New algorithms for decoding in the rank metric and an attack on the LRPC cryptosystem

Adrien Hauteville [#*1], Jean-Pierre Tillich [*2]

[#] *Université de Limoges, XLIM-DMI 123, Av. Albert Thomas, 87060 Limoges, Cedex, France*

[*] *Inria, Domaine de Voluceau, BP 105, Le Chesnay 78153, France*

[1] adrien.hauteville@etu.unilim.fr

[2] jean-pierre.tillich@inria.fr

April 22, 2015

Abstract

We consider the decoding problem or the problem of finding low weight codewords for rank metric codes. We show how additional information about the codeword we want to find under the form of certain linear combinations of the entries of the codeword leads to algorithms with a better complexity. This is then used together with a folding technique for attacking a McEliece scheme based on LRPC codes. It leads to a feasible attack on one of the parameters suggested in [11].

1 Introduction

McEliece schemes. The hardness of the problem of decoding a linear code makes its use very attractive in the cryptographic setting. Indeed it has been proven to be NP-complete for the Hamming metric in the seminal paper of Berlekamp, McEliece and van Tilborg [2]. Moreover, despite some significant research efforts, only exponential algorithms are known for it and the exponent has decreased only very slowly over time [1]. One of the very first public-key cryptosystem [19] is actually (partly) based on this problem. It still belongs to the very few public key cryptosystems which remain unbroken today.

One of the drawbacks of this scheme is its large public key size. It relies on a particular code family, namely Goppa codes, which in many respects look like random linear codes but still have an efficient decoding algorithm. Since then, many approaches have been tried to reduce the key size: (i) alternative code families have been proposed, (ii) using codes with a large automorphism group such as quasi-cyclic codes, (iii) changing the metric used for the code and the code itself.

McEliece schemes based on rank metric codes. In this paper we focus on a proposal which is a mixture of the approaches (ii) and (iii): the LRPC scheme of [11]. It relies on a new family of codes, called Low Rank Parity Check (LRPC in short) codes which are devised for the rank metric. The first McEliece scheme based on rank metric codes was the Gabidulin-Paramonov-Tretjakov cryptosystem [9]. It relies on an analogue of Reed-Solomon codes for the rank metric, the "Gabidulin codes". The scheme got broken by Overbeck in [22]. One of the main reasons for its insecurity can be traced back to its rich algebraic structure. This is not the case for the LRPC scheme. For this family of codes, like for the MDPC codes based McEliece scheme of [20], it seems that key security and message security really rely on the same problem, namely finding a low rank weight (or moderate Hamming weight for [20]) codeword in a linear code with no structure.

Decoding for the rank metric. It is essential with this approach to have a good assessment of the complexity of solving the decoding problem in the rank metric. Recall that in Delsarte's language [5], linear rank metric codes are viewed as the subspace generated by a set of matrices of a same size over some finite field $\mathbb{F}_q$.

The associated decoding problem is also known under the name "MinRank" and is known to be NP-complete [3]. Generally such codes arise in the form of linear codes defined over some extension field $\mathbb{F}_{q^m}$.

This problem has attracted some attention in the cryptographic community and algorithms of exponential complexity have been devised for it [4, 21, 12, 14, 6].

Attacks on quasi-cyclic codes by folding the code. The parameters of the LRPC scheme have been devised in order to be safe against the aforementioned algorithms for decoding in the rank metric. However, the authors of the scheme have also used quasi-cyclic versions of such codes in order to reduce further the size of the parameters. It has been found out recently [7, 8] that McEliece versions based on quasi-cylic or quasi-monoidic codes can be attacked by reducing the size of the code by adding coordinates which belong to the same orbit of the automorphism group. This is called the "folding" process in these papers. When this process is applied to quasi-cyclic or quasi-dyadic alternant or Goppa codes suggested in the cryptographic community, this results in a much smaller alternant or Goppa code and this can be used to mount a key recovery attack. This approach was further investigated and the folding process was generalized by using a polynomial formalism in [17]. It was shown there that this approach can be used for the quasi-cyclic LRPC codes of [11] and gives a LRPC code of much smaller size but which still has in its dual low weight codewords. The decoding algorithm of [12] can then be used to find these low weight codewords in a more efficient way than for the original code. This results in a multiplicative gain in the complexity of the attack of order 2^{12} for one of the parameters proposed in [11].

Our contribution. Our contribution in the paper is threefold. First we show how certain rank decoding algorithms of [21, 12] may benefit from some partial knowledge on the codeword which is sought. We consider here that we are given certain linear combinations of the entries of the codeword. This generalizes the $\mathbb{F}_{q^m}$ linear case where a certain entry can be assumed to be equal to 1. Roughly speaking, when we search in the latter case for a rank weight w codeword using the algorithm of [21, 12] we have algorithms of complexity $q^{(w-1)\alpha}$ where α is some quantity that depends on the algorithm which is considered and some code parameters. We show how the complexity of these algorithms can be reduced to $q^{(w-a)\alpha}$ when we know a independent linear combinations of the code positions. We also obtain by the approach of [12] applied to the transposed code an algorithm with the same complexity as [21] but which is significantly simpler. Finally, we show that when the folding process is applied to the quasi-cyclic $\mathbb{F}_{q^m}$ linear codes considered in [11] we know two independent linear combinations of the codeword we are looking for, instead of just one. This is then used together with the generalized folding process of [17] to give a much more efficient attack than in [17].

2 Generalities about rank metric codes

Let us start with the definition of a matrix code

Definition 2.1 (Matrix code)**.** *A matrix code of size $m \times n$ over $\mathbb{F}_q$ is a linear code generated by matrices of size $m \times n$ over $\mathbb{F}_q$. When the code is of dimension K we say that it is an $[m \times n, K]$ matrix code over $\mathbb{F}_q$.*

Remark 2.2. *It will be convenient to express K under the form $K = k.m$. Notice that k is not necessarily an integer.*

It might be thought that this is nothing but a linear code of length $m.n$. The point of this definition is that we equip such codes with the rank metric

that is defined by $d(\boldsymbol{A}, \boldsymbol{B}) = \mathrm{Rank}(\boldsymbol{A} - \boldsymbol{B})$. The weight $|\boldsymbol{c}|$ of a word $\boldsymbol{c}$ is taken with respect to the rank, that is $|\boldsymbol{c}| \stackrel{\text{def}}{=} d(\boldsymbol{c}, 0) = \mathrm{Rank}(\boldsymbol{c})$. Generally such codes are obtained from $\mathbb{F}_{q^m}$ linear codes as follows

Definition 2.3 (Matrix code associated to an $\mathbb{F}_{q^m}$ linear code)**.** *Let $\mathcal{C}$ be an $[n, k]$ linear code over $\mathbb{F}_{q^m}$ and let $(\beta_1 \dots \beta_m)$ be a basis of $\mathbb{F}_{q^m}$ over $\mathbb{F}_q$. Each word $\boldsymbol{c} \in \mathcal{C}$ can be represented by an $m \times n$ matrix $\boldsymbol{M}(\boldsymbol{c}) = (M_{ij})_{\substack{1 \le i \le m \\ 1 \le j \le n}}$ over $\mathbb{F}_q$, with $c_j = \sum_{i=1}^{m} M_{ij}\beta_i$. The set $\{\boldsymbol{M}(\boldsymbol{c}), \boldsymbol{c} \in \mathcal{C}\}$ is the matrix code associated to the $\mathbb{F}_{q^m}$ linear code $\mathcal{C}$. It is of type $[m \times n, k.m]$.*

This definition depends of course of the basis chosen for $\mathbb{F}_{q^m}$. However changing the basis does not change the distance between codewords. The point of defining matrix codes in this way is that they have a more compact description. It is readily seen that an $[m \times n, k.m]$ matrix code can be specified from a systematic generator matrix by $k(n-k)m^2 \log_2 q$ bits whereas a $\mathbb{F}_{q^m}$-linear code uses only $k(n-k)\log_2 q^m = k(n-k)m \log_2 q$ bits. This is particularly interesting for cryptographic applications where this notion is directly related to the public key size. We can now define the two central problem in this field, namely

Problem 2.1 (Decoding in the rank metric)**.** *For a given matrix code $\mathcal{C}$ of type $[m \times n, K]$ over $\mathbb{F}_q$, a matrix $\boldsymbol{A}$ in $\mathbb{F}_q^{m \times n}$ and an integer w, find a codeword $\boldsymbol{c}$ in $\mathcal{C}$ such that* $\mathrm{Rank}(\boldsymbol{A} - \boldsymbol{c}) = w$.

Problem 2.2 (Low rank codeword problem)**.** *For a given matrix code $\mathcal{C}$ and an integer w, find a codeword $\boldsymbol{c}$ of rank weight w in $\mathcal{C}$.*

The decoding problem reduces to the low rank codeword problem by finding a codeword of weight w in the matrix code $\mathcal{C}'$ where $\mathcal{C}'$ is generated by the codewords of $\boldsymbol{C}$ and $\boldsymbol{A}$ when $\mathcal{C}$ does not contain codewords of rank weight w. In other words, decoding an error of weight w in an $[m \times n, K]$ matrix code reduces to the problem of finding a codeword of weight w in an $[m \times n, K+1]$ matrix code. Notice that the low rank codeword problem is slightly simpler for matrix codes obtained from $\mathbb{F}_{q^m}$ linear codes. Indeed, we may assume that the codeword $\boldsymbol{c}$ of weight w contains a coordinate equal to 1. This follows from the fact that multiplying $\boldsymbol{c}$ by any nonzero element of $\mathbb{F}_{q^m}$ does not change the rank of the associated matrix. In other words, we have some additional knowledge about the codeword (or the error) of weight w in this case. Notice that the support trapping decoding algorithm (see next section) of [12] and the decoding algorithm of [21] given for $\mathbb{F}_{q^m}$ both exploit this knowledge. They have an asymptotic exponential complexity of the form $q^{\alpha(w-1)}$ whereas it would have been only $q^{\alpha w}$ for an unstructured matrix code with the same parameters.

3 A support trapping decoding algorithm

[12] has introduced a very neat and simple algorithm for decoding in the rank metric. It can be considered as a support trapping decoding algorithm for an $[m \times n, k.m]$ matrix code that tries to guess a subspace F of the column space $\mathbb{F}_q^m$ of $m \times n$ matrices over $\mathbb{F}_q$ that contains the column space E of the error $\boldsymbol{e}$ we want to find. Since we focus on the low-weight finding problem in this article

we will explain this algorithm in the case we look for a codeword of weight w in an $[m \times n, K]$ code. In this case, E is the column space of $\boldsymbol{c}$. The next step is then to express the columns $\boldsymbol{c}_i$ of $\boldsymbol{c}$ in a basis $\boldsymbol{f}_1, \ldots, \boldsymbol{f}_r$ of F, that is $\boldsymbol{c}_i = \sum_{j=1}^{r} x_{ij} \boldsymbol{f}_j$. This gives $n.r$ unknowns (the x_{ij}'s). From a parity-check matrix of the matrix code we deduce $n.m - k.m = (n-k)m$ equations involving the entries of $\boldsymbol{c}$ that can all be expressed in terms of the x_{ij}'s. In other words, we have a linear system with $(n-k)m$ equations and $n.r$ unknowns. We choose r to be the least integer such that the number of unknowns is less than the number of equations. In our case, $r = m - \left\lceil \frac{km}{n} \right\rceil$.

The complexity of the algorithm depends on the probability of having $E \subset F$. It is equal to the number of subspaces of dimension w in a subspace of dimension r, divided by the number of all subspaces of dimension w in $\mathbb{F}_q^m$. This probability can be easily expressed with Gaussian coefficients, which counts the number of subspaces of a vector-space :

$$p = \frac{\begin{bmatrix} r \\ w \end{bmatrix}_q}{\begin{bmatrix} m \\ w \end{bmatrix}_q} = \Theta\left(q^{-w(m-r)}\right) \tag{1}$$

We use here the following notation.

Notation 3.1. $\begin{bmatrix} m \\ w \end{bmatrix}_q$ *is the Gaussian binomial coefficient that is equal to the number of subspaces of* $\mathbb{F}_q^m$ *of dimension* w*. Recall that this coefficient satisfies* $\begin{bmatrix} m \\ w \end{bmatrix}_q = \Theta\left(q^{w(m-w)}\right)$.

The cost to solve a linear system of $(n-k)m$ unknown by Gaussian elimination is $\mathcal{O}\big((n-k)^3 m^3\big)$. Thus, the overall expected complexity for this algorithm is $\mathcal{O}\big((n-k)^3 m^3 q^{w\lceil \frac{km}{n} \rceil}\big)$.

As explained in [12] F can be viewed as the support of a codeword for the rank metric. What makes this notion interesting is that it establishes a parallel with the Hamming metric : indeed, if we know the support $\boldsymbol{c}$ of a codeword $\boldsymbol{c}$ we can recover $\boldsymbol{c}$ in polynomial time by solving a linear system.

This algorithm is much more efficient than the algorithm in [21] when $m \leq n$. Let us notice that in the case $m > n$ we can improve this algorithm in a simple way by using the notion of the transposed code which is defined as follows [10]

Definition 3.2 (transposed code)**.** *The transposed of an* $[m \times n, K]$ *matrix code* $\mathcal{C}$ *over* $\mathbb{F}_q$ *is a* $[n \times m, K]$ *matrix code* $\mathcal{C}^T$ *over* $\mathbb{F}_q$ *obtained by* $\mathcal{C}^T = \{\boldsymbol{M}^T, \boldsymbol{M} \in \mathcal{C}\}$.

The idea underlying the definition of such a code is that transposing a matrix preserves its rank, therefore finding the minimum rank weight (nonzero) codeword $\mathcal{C}$ can be obtained from the transpose of the minimum rank weight (nonzero) codeword of $\mathcal{C}^T$. Notice that taking the transpose basically swaps the role of n and m. This notion can be used when $m \geq n$ for finding a codeword of weight w in a matrix code $\mathcal{C}$ by looking for a codeword of weight w in $\mathcal{C}^T$. It is readily seen that this leads to an

algorithm of complexity $\mathcal{O}\big((n-k)^3 m^3 q^{w\lceil k\rceil}\big)$ for finding a codeword of weight w. This is precisely the complexity that the algorithm of [21] would give for finding a codeword of weight w in a matrix code. However the algorithm presented here is much simpler than the algorithm of [21].

4 A low weight codeword finding algorithm using additional knowledge on the codeword

In this section, we assume that we have additional knowledge about the codeword of weight w we want to find in the form of linear combinations of its columns. More precisely we are looking for an algorithm whose input and output are specified in Algorithm 1.

Algorithm 1: Low rank codeword finding with additional information

Input :
(i) an $[m \times n, k.m]$ matrix code $\mathcal{C}$ over $\mathbb{F}_q$ that has at least one codeword $\boldsymbol{c} = (c_{ij})_{\substack{1\le i\le m\\ 1\le j\le n}}$ of rank weight w
(ii) a elements $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_a$ in $\mathbb{F}_q^m$ that are linear combinations of columns of $\boldsymbol{c}$.
(iii) the coefficients λ_{ij}'s of these linear combinations, that is if we denote by $\boldsymbol{c}_{.,j} = (c_{ij})_{1\le i\le m}$ the j-th column of $\boldsymbol{c}$, then $\boldsymbol{c}'_i = \sum_{j=1}^{n} \lambda_{ij} \boldsymbol{c}_{.,j}$ for $i \in \{1, \ldots, a\}$.
Assumes: $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_a$ are linearly independent.
Output : a codeword of $\mathcal{C}$ of rank weight w.

The case of a matrix code obtained from an $\mathbb{F}_{q^m}$-linear code is a particular case of such an additional knowledge: as explained before we can assume that one of the columns of the codeword we are looking for is the column $(10\ldots0)^T$. The folding attack that we present in Section 5 will provide another example where we have the knowledge of two independent linear combinations of the columns and will use in an essential way the algorithm we give here.

4.1 The case $n \geq m$

We use here a variation of the support trapping algorithm [12]. The case when $a = 1$ and when the matrix code is obtained from an $\mathbb{F}_{q^m}$-linear code is already treated in [12, Prop. 3.1]. Generalizing this argument to the more general setting considered here just consists in chosing in the error trapping algorithm recalled in Section 3 an F as a random subspace of dimension r that contains the subspace generated by the a elements $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_a$. This leads to the following proposition.

Proposition 4.1. *The support trapping algorithm outlined above has expected complexity* $\mathcal{O}\big((n-k)^3 m^3 q^{(w-a)\lceil \frac{km}{n}\rceil}\big)$ *when applied on a matrix code over* $\mathbb{F}_q$ *of type* $[m \times n, k.m]$.

The complexity given follows almost immediately from the following proposition.

Proposition 4.2. *Let E be a subspace of dimension w of $\mathbb{F}_q^m$ and let E' be a subspace of E of dimension a. Let S be the set of subspaces of dimension r of $\mathbb{F}_q^m$ that contain E' and let F be an element of S chosen uniformly at random. We have*

$$\text{Prob}\left(E \subset F\right) = \frac{\begin{bmatrix} r-a \\ w-a \end{bmatrix}_q}{\begin{bmatrix} m-a \\ w-a \end{bmatrix}_q} = \Theta\left(q^{(w-a)(m-r)}\right).$$

Proof. Let $V = \mathbb{F}_q^m / E' \simeq \mathbb{F}_q^{m-a}$.
Let π be the canonical surjection from $\mathbb{F}_q^m$ to V :

$$\begin{array}{rccc} \pi: & \mathbb{F}_q^m & \to & V \\ & \boldsymbol{x} & \mapsto & \boldsymbol{x} + E' \end{array}$$

It is well known that π gives a one-to-one correspondence between the subspaces of $\mathbb{F}_q^m$ which contain E' and the subspaces of V.

Lemma 4.3. *Let F be a subspace of dimension r of $\mathbb{F}_q^m$ that contains E'. The dimension of $\pi(F)$ is $r-a$.*

Proof. Let $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_a)$ be a basis of E'. We can complete this basis into a basis $(\boldsymbol{e}_1, \ldots, \boldsymbol{e}_a, \boldsymbol{f}_1, \ldots, \boldsymbol{f}_{r-a})$ of F. It is obvious that $(\pi(\boldsymbol{f}_1), \ldots, \pi(\boldsymbol{f}_{r-a}))$ is a basis of $\pi(F)$, so $\dim \pi(F) = r-a$ □

By using this lemma, we finish the proof of Proposition 4.2. There are $\begin{bmatrix} m-a \\ w-a \end{bmatrix}_q$ subspaces of V of dimension $w-a$. This implies that there exist $\begin{bmatrix} m-a \\ w-a \end{bmatrix}_q$ subspaces of dimension w of $\mathbb{F}_q^m$ that contain E'.

Let F be a subspace of dimension r of $\mathbb{F}_q^m$ that contains E'. According to the previous lemma, $\dim \pi(F) = r-a$. So $\pi(F)$ contains $\begin{bmatrix} r-a \\ w-a \end{bmatrix}_q$ subspaces of dimension $w-a$. From this, we deduce that F contains $\begin{bmatrix} r-a \\ w-a \end{bmatrix}_q$ subspaces of dimension w that contain E'. Hence

$$\text{Prob}\left(E \subset F\right) = \frac{\begin{bmatrix} r-a \\ w-a \end{bmatrix}_q}{\begin{bmatrix} m-a \\ w-a \end{bmatrix}_q} = \Theta\left(q^{-(w-a)(m-r)}\right)$$

□

The proof of Proposition 4.1 follows directly from this Proposition. Indeed we choose in the support trapping algorithm, E' to be the linear space generated by $\boldsymbol{c}'_1, \ldots, \boldsymbol{c}'_a$ and F as a random subspace of $\mathbb{F}_q^m$ that contains E'. The expected complexity of the support trapping algorithm is now given by the inverse of the probability that we computed in Proposition 4.1 multiplied by the complexity of solving a linear system with $(n-k)m$ equations.

4.2 The case $m > n$

This will be treated essentially by a variation on the error trapping algorithm applied to the transposed code which uses in a suitable way the additional knowledge about the codeword we want to find. The technical difficulty we face here can be described as follows. If we had additional knowledge about $\boldsymbol{c}$ in the form of a independent elements belonging to the row space of $\boldsymbol{c}$, then we could immediately apply the algorithm given in Section 3 to the transposed code. However it turns out that in the case we are interested in, the knowledge about $\boldsymbol{c}$ that we have concerns the column space of $\boldsymbol{c}$. In this case, when we transpose $\boldsymbol{c}$ to reverse the role of n and m, this translates into some knowledge of the row space of $\boldsymbol{c}^T$ and we can not use the algorithm of Section 3 anymore. This is why we are going to consider a slightly more complicated algorithm which is able to use some knowledge on the column space of $\boldsymbol{c}$. It will be essential for our attack that is given in Section 5 to work to have an efficient algorithm for finding low-rank codewords by exploiting some knowledge about the low-rank codeword we are looking for. Even if the underlying code is defined for $m < n$ it turns out that we are reducing this problem to another low-rank finding problem in a new code where $m > n$. Of course we could still use the algorithm described in Section 3. It appears that the Ourivski-Johansson [21] is better in the regime when $n < m$. However, this algorithm in its [21] form is unable to take full advantage of the knowledge we have about the low-weight codeword we are looking for. It would have been possible to give a version of the Ourivski-Johansson that exploits additional knowledge in the same way we generalized slightly the support trapping algorithm of [12]. However, the Ourivski-Johannson algorithm is rather involved and we will use another approach here that recovers the same work factor as the Ourivski-Johannsson algorithm in the case of decoding a $\mathbb{F}_{q^m}$-linear code but in a much simpler fashion. This new decoding algorithm is in essence a support trapping algorithm working on the transposed code. It will also be able to use in a simple way additional knowledge about the low rank word we are looking for.

The point is now that by applying a version of the support trapping algorithm of [12] that makes use in a suitable way of the additional knowledge we have about the support, we basically recover an algorithm with the same complexity as the Ourivski-Johansson algorithm for decoding $\mathbb{F}_{q^m}$ linear codes. More generally it will have an exponential asymptotic complexity of order $\mathcal{O}\big((n-k)^3 m^3 q^{(w-a)k}\big)$ for an $[m \times n, k.m]$ matrix code over $\mathbb{F}_q$ when we know a independent linear combinations of the columns of the matrix codeword of rank w we are looking for.

This algorithm can be described as follows

Step 1 (transformation of the code): We first transform the matrix code $\mathcal{C}$ by multiplying it at the right by an $n \times n$ invertible matrix $\boldsymbol{P}$ such that $\boldsymbol{c}$ gets transformed in a matrix $\boldsymbol{c}'$ whose i first columns are precisely the $\boldsymbol{c}'_i$'s defined before. In other words, we consider the code $\mathcal{C}' = \mathcal{C}\boldsymbol{P}$. If $\boldsymbol{c}$ is a word of rank weight w then $\boldsymbol{c}'$ is still a word of rank weight w. Moreover by assumption on the independence of the $\boldsymbol{c}'_i$'s for $i \in \{1, \ldots, a\}$ we can further multiply $\mathcal{C}'$ on the left by an $m \times m$ invertible matrix $\boldsymbol{Q}$ such that $\boldsymbol{c}'$ gets transformed in a matrix $\boldsymbol{c}$" whose first a columns are the first a elements $\boldsymbol{e}_1, \ldots, \boldsymbol{e}_a$ of the canonical basis of $\mathbb{F}_q^m$, that is $\boldsymbol{e}_i$ has only zero entries with the exception of the i-th entry which

is equal to 1. Let $\mathcal{C}$" be the resulting code obtained by these operations, that is

$$\mathcal{C}" = \boldsymbol{Q}\mathcal{C}\boldsymbol{P}$$

Notice that $\boldsymbol{c}$" has still rank w.

Step 2: (setting up the unknowns of the linear system) We are now basically going to apply a variation of the support trapping algorithm of [12] on $\mathcal{C}"^T$ by choosing a subspace V of $\mathbb{F}_q^n$ of dimension r (r will be specified later on) for which we hope that it contains the subspace generated by the columns of $\boldsymbol{c}"^T$. A basis $\boldsymbol{v}_1, \ldots, \boldsymbol{v}_r$ of this space is chosen such that

$$\begin{aligned} v_{j,i} &= 0 \text{ for } i, j \text{ in } \{1, \ldots, a\} \text{ and } i \neq j & (2)\\ v_{i,i} &= 1 \text{ for } i \text{ in } \{1, \ldots, a\} & (3)\\ v_{j,i} &= 0 \text{ for } i \text{ in } \{a+1, \ldots, r\} \text{ and } j \text{ in } \{1, \ldots, a\} & (4) \end{aligned}$$

where $v_{j,i}$ denotes the j-th coordinate of $\boldsymbol{v}_i$. The entries $v_{j,i}$ are chosen uniformly at random for i in $\{a+1, \ldots, r\}$ and j in $\{a+1, \ldots, n\}$. The entries of $v_{j,i}$ for i in $\{1, \ldots, a\}$ and j in $\{a+1, \ldots, n\}$ will be chosen afterwards. Denote by $\boldsymbol{C}_1, \ldots, \boldsymbol{C}_m$ the m columns of $\boldsymbol{c}"^T$. Let us introduce the $x_{s,t}$'s in $\mathbb{F}_q$ that are such that

$$\boldsymbol{C}_s = \sum_{t=1}^{r} x_{s,t}\boldsymbol{v}_t \text{ for } s \text{ in } \{1, \ldots, m\}. \quad (5)$$

Notice now the following point

Lemma 4.4. *For $s > a$ and all i in $\{1, \ldots, a\}$ we have $x_{s,i} = 0$. If we denote by $C_{i,j}$ the i'th element of the j-th column $\boldsymbol{C}_j$ of $\boldsymbol{c}"^T$ then $C_{i,j} = 0$ for all i, j in $\{1, \ldots, a\}$ with the exception of the diagonal elements $C_{i,i}$ that are equal to* 1.

Proof. Denote by $\boldsymbol{R}_1, \ldots, \boldsymbol{R}_n$ the n rows of $\boldsymbol{c}"^T$. Notice that

$$\boldsymbol{R}_i = \boldsymbol{e}_i, \text{ for } i \text{ in } \{1, \ldots, a\} \quad (6)$$

where the $\boldsymbol{e}_i$'s are as before the canonical basis of $\mathbb{F}_q^m$. This implies directly that $C_{i,j} = 0$ for all i, j in $\{1, \ldots, a\}$ with the exception of the diagonal elements $C_{i,i}$ that are equal to 1. Moreover, by using (6) together with (2),(3) and (4) we know that $x_{s,i} = 0$ for $s > a$ and all i in $\{1, \ldots, a\}$. □

This motivates to define as unknowns the $(m-a)(r-a)+a(n-a)$ quantities $x_{s,t}$ and $C_{i,j}$ for s in $\{a+1, \ldots, m\}$, t in $\{a+1, \ldots, r\}$, i in $\{a+1, \ldots, n\}$ and j in $\{1, \ldots, a\}$.

Moreover these unknowns satisfy $nm - km = (n-k)m$ linear equations obtained from the fact $\boldsymbol{c}"^T$ belongs to $\mathcal{C}"^T$ which is a matrix code of dimension km. They can be obtained by computing a parity-check matrix of this code, then expressing the linear equations that the entries of $\boldsymbol{c}"^T$ have to satisfy and then replacing these entries by the aforementioned unknowns by using (5) and Lemma 4.4. We choose r such that the number of equations, that is $(n-k)m$ is at least equal to the number of unknowns, that is

$$(n-k)m \geq (m-a)(r-a) + a(n-a)$$

This can be obtained by choosing

$$r \stackrel{\text{def}}{=} \left\lfloor \frac{m}{m-a}(n-k) + a\frac{m-n}{m-a} \right\rfloor$$

Step 3: (solving the linear system) The last point just consists in solving the linear system, this yields $\boldsymbol{c}"^T$ and from this we deduce $\boldsymbol{c}"$ and then $\boldsymbol{c}$ by

$$\boldsymbol{c} = \boldsymbol{Q}^{-1}\boldsymbol{c}"\boldsymbol{P}^{-1}$$

The last point to understand is under which condition V contains the subspace generated by the columns of $\boldsymbol{c}"^T$. This depends on how we specify the entries $v_{j,i}$ for i in $\{1,\ldots,a\}$ and j in $\{a+1,\ldots,n\}$. We choose them such that (5) is verified for s in $\{1,\ldots,a\}$. This can obviously be done by choosing

$$\boldsymbol{v}_i = \boldsymbol{C}_i \text{ for } i \in \{1,\ldots,a\} \tag{7}$$

Lemma 4.5. *Let V be chosen by a basis $\boldsymbol{v}_1,\ldots,\boldsymbol{v}_r$ such that its a first elements are given by (7) and as specified in Step 2 for the other elements. Let W be the subspace generated by the columns of $\boldsymbol{c}"^T$. Let W_0 be the subspace of W that is formed by the elements whose first a entries are all equal to 0. In the same way, we denote by V_0 the subspace that is formed by the elements of V whose first a entries are all equal to 0. We have $W \subset V$ iff $W_0 \subset V_0$.*

Proof. It is clear that $W \subset V$ implies $W_0 \subset V_0$.

Now assume that $W_0 \subset V_0$. Notice that W is generated by W_0 and by the first a columns of $\boldsymbol{c}"$, that is $\boldsymbol{C}_1,\ldots,\boldsymbol{C}_a$. Since V is generated by the same first a columns of $\boldsymbol{c}"$, $\boldsymbol{C}_1,\ldots,\boldsymbol{C}_a$ and by V_0 we have that $W \subset V$. □

Putting all these considerations together we obtain that

Theorem 4.6. *Let $\mathcal{C}$ be an $[m \times n, k.m]$ matrix code which has at least one codeword of rank weight w for which we know a independent linear combinations of its columns as specified in Algorithm 1. Assume that $n \leq m$ and let $r \stackrel{def}{=} \left\lfloor \frac{m}{m-a}(n-k) + a\frac{m-n}{m-a} \right\rfloor$. Then the algorithm given in this section outputs a codeword of weight w with complexity $\mathcal{O}\big((n-k)^3 m^3 q^{(w-a)(n-r)}\big)$.*

Proof. This follows immediately from Lemma 4.5 and Proposition 4.2 that show that we will try an expected number of $\dfrac{\begin{bmatrix} n-a \\ w-a \end{bmatrix}_q}{\begin{bmatrix} r-a \\ w-a \end{bmatrix}_q} = \Theta\left(q^{(w-a)(n-r)}\right)$ spaces V before finding the right one if there is only one codeword $\boldsymbol{c}$ which has the right form. This is of course an upper bound if there are more than one codeword that have the right form. Each try of a tentative space V takes time $\mathcal{O}\big((n-k)^3 m^3\big)$ whose complexity is dominated by Step 3 when we solve a linear system with $(n-k)m$ equations and a number of unknowns that is less than the number of equations. □

5 Folding and projecting attack

In this section we present a key recovery attack on the LRPC cryptosystem [11]. The codes used there are defined by

Definition 5.1 (LRPC code)**.** *Let C be the matrix code associated to the $\mathbb{F}_{q^m}$-linear code with a full rank parity check matrix $\boldsymbol{H}$ of size $(n-k)\times n$. It defines an $[n,k]$ LRPC of weight d if the $\mathbb{F}_q$ subspace of $\mathbb{F}_{q^m}$ generated by the entries of $\boldsymbol{H}$ is of dimension d.*

A probabilistic decoding algorithm of polynomial time for LRPC codes is presented in [11]. This algorithm uses in an essential way that such a code has a parity-check matrix $\boldsymbol{H}$ that has entries in a subspace of small dimension. $\boldsymbol{H}$ can be easily hidden by giving a systematic parity-check matrix $\boldsymbol{H}_{\text{syst}}$. This family of codes can then be used in a McEliece type scheme [19] : the secret key is $\boldsymbol{H}$ and the public key is $\boldsymbol{H}_{\text{syst}}$. To recover the secret key, the attacker must find a word of weight d in the dual of $\mathcal{C}$, which is hard in principle. To decrease the key sizes, double-circulant LRPC codes are suggested in [11].

Definition 5.2. *A double-circulant LRPC (DC-LRPC) code of weight d over $\mathbb{F}_{q^m}$ is an LRPC code defined from a double-circulant parity check matrix $\boldsymbol{H} = \begin{pmatrix}\boldsymbol{H}_1 & \boldsymbol{H}_2\end{pmatrix}$ where $\boldsymbol{H}_1$ and $\boldsymbol{H}_2$ are two circulant matrices and the $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ generated by the entries of $\boldsymbol{H}$ is of dimension d.*

5.1 Folded and projected codes

We present here two new ingredients of the attacks that follow, namely the notion of folded code and the notion of projected code. The first attack uses only folding but the second attack uses both. The notion of projected codes uses the polynomial framework for dealing with quasi-cyclic codes [16, 15]. Quasi-cyclic codes are a generalization of double-circulant codes : they are defined by a parity check matrix formed only from circulant blocks. Such a quasi-cyclic code of length $N = \ell n$ defined over a finite field $\mathbb{K}$, where the size of the circulant blocks is n, can also be viewed as code over the ring $\mathbb{K}[X]/(X^n-1)$. This is a specific instance of cellular codes that are codes defined over a ring $\mathcal{R} = \mathbb{K}[X]/(f(X))$ where $f(X)$ is a polynomial of $\mathbb{K}[X]$.

For the reader's convenience, we recall here the polynomial formalism of [18, 15] and follow the presentation given in [17]. Recall that the fact that quasi-cyclic codes can be viewed as codes defined over the ring $\mathbb{K}[X]/(f(x))$ follows directly from

Proposition 5.3. *The set of circulant matrices of size $n\times n$ over $\mathbb{F}_{q^m}$ is isomorphic to the $\mathbb{F}_{q^m}$-algebra $\mathbb{F}_{q^m}[X]/(X^n-1)$ by the function ϕ*

$$\phi\left(\sum_{i=0}^{n-1} a_i X^i\right) = \begin{pmatrix} a_0 & a_1 & \dots & a_{n-1} \\ a_{n-1} & a_0 & \dots & a_{n-2} \\ & \ddots & \ddots & \\ a_1 & a_2 & \dots & a_0 \end{pmatrix}$$

More generally we consider codes over a finite field $\mathbb{K}$ derived from codes defined over a ring

$$\mathcal{R} \stackrel{\text{def}}{=} \mathbb{K}[X]/(f(X))$$

where f is some polynomial in $\mathbb{K}[X]$ of degree n. They are derived from the following $\mathbb{K}$-isomorphism $\psi : \mathcal{R} \to \mathbb{K}^n$:

$$a(X) = \sum_{i=0}^{n-1} a_i X^i \mapsto \psi(a(X)) = (a_0, \ldots, a_{n-1}).$$

They are called cellular codes and are defined by

Definition 5.4 (cellular code)**.** *Consider a submodule M of $\mathcal{R}^\ell$ of rank s. Let $\psi^\ell : \mathcal{R}^\ell \to \mathbb{K}^{\ell n}$ that maps an element $(f_1, \ldots, f_l)$ of $\mathcal{R}^\ell$ to $\mathbb{K}^{\ell n}$ by mapping each f_i to $\psi(f_i)$. The cellular code associated to M is given by $\psi^\ell(M)$. It is said to have index ℓ and it is a $\mathbb{K}$-linear code of length ℓn.*

Remark 5.5. *In order to avoid cumbersome notation, we identified M with $\psi^\ell(M)$ in Section 5. Sometimes it will better to view the cellular code $\psi^\ell(M)$ as M and we will freely do this.*

To obtain a generator matrix of the cellular code from a generator matrix

$$\boldsymbol{G}_M = \begin{pmatrix} a_{1,1}(X) & \ldots & a_{1,\ell}(X) \\ \vdots & \ddots & \vdots \\ a_{s,1}(X) & \ldots & a_{s,\ell}(X) \end{pmatrix}$$

of the rank s-submodule we introduce the following mapping $\phi : \mathcal{R} \to \mathbb{K}^{n \times n}$:

$$a(X) = \sum_{i=0}^{n-1} a_i X^i \mapsto \phi(a(X)) = \begin{pmatrix} \psi(a(X)) \\ \psi(Xa(X)) \\ \vdots \\ \psi(X^{n-1}a(X)) \end{pmatrix}$$

It is a bijective morphism of $\mathbb{K}$-algebras. When $f(X) = X^n - 1$ this is precisely the mapping that appears in Proposition 5.3. A generator matrix of the associated cellular code is now given by

$$\boldsymbol{G} = \begin{pmatrix} \boldsymbol{A}_{1,1} & \ldots & \boldsymbol{A}_{1,\ell} \\ \vdots & \ddots & \vdots \\ \boldsymbol{A}_{s,1} & \ldots & \boldsymbol{A}_{s,\ell} \end{pmatrix}$$

where $\boldsymbol{A}_{i,j} = \phi(a_{i,j}(X))$. This implies that the dimension k of the cellular code satisfies $k \leq ns$.

Definition 5.6 (projected code)**.** *Consider a cellular code $\mathcal{C}$ of index ℓ defined over $\mathcal{R} \stackrel{def}{=} \mathbb{K}[X]/(f(X))$ and let $g(X)$ be a divisor of $f(X)$ in $\mathbb{K}[X]$. The projected cellular code $\overline{\mathcal{C}}^g$ is obtained by viewing a codeword $\boldsymbol{c}$ of $\mathcal{C}$ as an element of R^l : $\boldsymbol{c} = (\boldsymbol{c}_1, \ldots, \boldsymbol{c}_\ell)$ and applying the surjective morphism Π from $\mathbb{K}[X]/(f(X))$ to $\mathbb{K}[X]/(g(X))$ defined by $\Pi(a(X)) = a(X) \pmod{g(X)}$ to every entry $\boldsymbol{c}_i$.*

In the particular case where $f(X) = X^n - 1$ and $g(X) = X^m - 1$ where m is a divisor of n, projecting corresponds to folding in the sense of [7, 8].

Definition 5.7 (folded code)**.** *Consider a quasi-cyclic code $\mathcal{C}$ of index ℓ and length $n\ell$. Let m be a divisor of n. Its folded code of order m is a quasi-cyclic code of index ℓ and length $m\ell$ obtained by mapping each codeword $\boldsymbol{c} = (c_0, \ldots, c_{n\ell-1})$ of $\mathcal{C}$ to the codeword $\boldsymbol{c}' = (c'_0, \ldots, c'_{m\ell-1})$ where*

$$c'_i = \sum_{s=0}^{\frac{n}{m}-1} c_{an+b+sm}$$

and a and b are the quotient and the remainder of the euclidean division of i by m:

$$i = am + b \text{ with } a \text{ and } b \text{ integer and } b \in \{0, \ldots, m-1\}.$$

This really amounts to sum the coordinates that belong to the same orbit of a (permutation) autorphism of order n/m that leaves the quasi-cyclic code invariant.

There are two points which make these two notions very interesting in the cryptographic setting. The first point is that these two reductions of the code do not lead to a trivial code at the end (one could have feared to end up with the full space after projecting or folding). This comes from the following proposition that is proved in [17, Corollary 1]

Proposition 5.8. *Consider a cellular code $\mathcal{C}$ of index ℓ defined over $\mathcal{R} \stackrel{def}{=} \mathbb{K}[X]/(f(X))$ and let $g(X)$ be a divisor of $f(X)$ in $\mathbb{K}[X]$. The length of the projected code $\overline{\mathcal{C}}^g$ is $\ell \deg g$ whereas the dimension of $\overline{\mathcal{C}}^g$ is less than or equal to $s\ell$ where s is the rank of the cellular code.*

The second point is that this operation of folding behaves nicely with respect to projecting a quasi-cyclic code defined over an extension field $\mathbb{F}_{q^m}$ with respect to the rank distance over $\mathbb{F}_q$ when the divisor g belongs to $\mathbb{F}_q[X]$

Proposition 5.9 ([17, Prop.3])**.** *Consider a cellular code $\mathcal{C}$ of index ℓ defined over $\mathcal{R} \stackrel{def}{=} \mathbb{F}_{q^m}[X]/(f(X))$ and let $g(X)$ be a divisor of $f(X)$ in $\mathbb{F}_q(X)$. Denote by Π the associated projection operation. We have*

$$\mathrm{Rank}(\Pi(\boldsymbol{c})) \leq \mathrm{Rank}(\boldsymbol{c})$$

for any $\boldsymbol{c} \in \mathcal{C}$ where we view these codewords as matrices in $\mathbb{F}_q^{m \times \ell \deg f}$ or in $\mathbb{F}_q^{m \times \ell \deg g}$ by taking the matrix form of these codewords as defined in Section 2.

Notice that this proposition can always be applied to folded codes. These two propositions allow to search for a codeword $\boldsymbol{c}$ of rank w in a quasi-cyclic code $\mathcal{C}$ of index ℓ and length $n\ell$ defined over $\mathbb{F}_{q^m}$ by projecting it with respect to a divisor of $X^n - 1$ that belongs to $\mathbb{F}_q[X]$ (or by folding it) and looking for a word of rank $\leq w$ in the projected or folded code. Roughly speaking, the first proposition ensures that we are not looking for a word w in the entire space. From the second proposition, we expect that as long w is below the Gilbert-Varshamov bound of the folded code, the codeword of weight $\leq w$ we will find in the projected code corresponds to the projection of $\boldsymbol{c}$. This allows to recover easily $\boldsymbol{c}$.

5.2 A first attack based on folding

Let $\mathcal{C}$ be a DC-LRPC $[2k, k]$ code of weight d over $\mathbb{F}_{q^m}$ obtained from a parity-check matrix $\boldsymbol{H}$. To recover $\boldsymbol{H}$ it is clearly sufficient to find a codeword of rank weight d in the dual $\mathcal{C}^\perp$ of $\mathcal{C}$. Let $\mathcal{C}'$ be the folding of order 1 of $\mathcal{C}^\perp$.

It is in general a $[2, 1]$ code. This folding reveals some additional information about the subspace F of $\mathbb{F}_{q^m}$ generated by the coefficients of H. We namely have

Proposition 5.10. *Let $\boldsymbol{c}' = (c'_1, c'_2)$ be in $\mathcal{C}'$. There exists $\boldsymbol{c}$ of weight d in $\mathcal{C}^\perp$ such that the $\mathbb{F}_q$-subspace generated by the coordinates of $\boldsymbol{c}$ contains c'_1 and c'_2.*

Proof. If $\mathcal{C}'$ is the all-zero code or $\boldsymbol{c}' = 0$ the conclusion follows directly.

Assume now that this is not the case. In this case, $\mathcal{C}'$ is of dimension 1. Consider a codeword $\boldsymbol{c}$ of $\mathcal{C}^\perp$ which is of weight d. Let $\boldsymbol{c}$" be the folded version of $\boldsymbol{c}$. We have in this case $\boldsymbol{c}' = \alpha\boldsymbol{c}$" for some $\alpha \in \mathbb{F}_{q^m}^\star$. Note that $\boldsymbol{c}'$ is the folded version of $\alpha\boldsymbol{c}$. We observe now two things and this finishes the proof

- $d = \mathrm{Rank}(\boldsymbol{c}) = \mathrm{Rank}(\alpha\boldsymbol{c})$ where we view these codewords as matrices in $\mathbb{F}_q^{m\times n}$ as explained in Section 2.
- c'_1 and c'_2 are in the $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ generated by the coordinates of $\alpha\boldsymbol{c}$.

□

We can use (c'_1, c'_2) in the decoding algorithm described in Section 4. From $\boldsymbol{c}$ we recover immediately a parity-check matrix of the form $\beta\boldsymbol{H}$, where $\beta \in \mathbb{F}_{q^m}\backslash\{0\}$, by building a parity-check matrix from $\boldsymbol{c}$ and its cyclic shifts. This gives an attack of complexity $\mathcal{O}\big(k^3m^3q^{(d-2)\lceil\frac{m}{2}\rceil}\big)$. However for the parameters proposed in [11, 13], this does not improve the attacks already considered there. However, this proposition together with another projection of the code will lead to a feasible attack against a certain parameter of [11, 13] as we now show.

5.3 An improved attack based on folding and projecting

To improve the attack, we search for a word of weight d in a projected code. This new attack depends on the factorization of $X^k - 1$. The length of the projected code we are interested in will be smaller than m and we will use the algorithm of Subsection 4.2 instead. The attack can be described as

Step 1: Compute $\mathcal{C}'$ the folding of order 1 of $\mathcal{C}^\perp$ and extract a codeword (c'_1, c'_2) in it.

Step 2: Compute the projected code $\overline{\mathcal{C}^\perp}^D$ with respect to a certain divisor $D(X)$ of $X^k - 1$ in $\mathbb{F}_q[X]$.

Step 3: Find a codeword $\boldsymbol{c}$" in $\overline{\mathcal{C}^\perp}^D$ of weight w such that the $\mathbb{F}_q$ space generated by its coordinates contains c'_1 and c'_2 by using the algorithm of Subsection 4.2.

Step 4: Let F be the $\mathbb{F}_q$-subspace of $\mathbb{F}_{q^m}$ generated by the coordinates of $\boldsymbol{c}$". Find the codeword $\boldsymbol{c}$ in $\mathcal{C}^\perp$ of rank weight w whose support is F (meaning that the $\mathbb{F}_q$-subspace generated by its coordinates should belong to F.)

What justifies the third step is the fact that Proposition 5.10 generalizes easily to the projected code, whereas what justifies Step 4 is the fact that it is

extremely likely that $\boldsymbol{c}$" is the projection of a codeword in $\mathcal{C}^\perp$ of weight d we are looking for. We recover in this case such a codeword by the process of Step 4. The complexity of this attack is dominated by the third step and is given by Theorem 4.6.

In [11], some parameters for the LRPC cryptosystem are suggested. They are recalled in the following table

n	k	m	q	d	security
74	37	41	2	4	80
94	47	47	2	5	128
68	34	23	2^4	4	100

In each case the factorization of $X^k - 1$ in $\mathbb{F}_q[X]$ is given by
(i) $X^{37} - 1 = (X-1)\sum_{i=0}^{36} X^i$
(ii) $X^{47} - 1 = (X-1)PQ$ with $\deg P = \deg Q = 23$
(iii) $X^{34} - 1 = (X-1)^2(P_1 \dots P_8)^2$ with $\deg P_i = 2$, for all $i \in [\![1; 8]\!]$.

In the first case, the polynomial $X^{37}-1$ has only two divisors, so we can only use the first attack. In the second case, we can choose $D = P$ or Q to obtain a folded code of dimension 23. According to Theorem 4.6, the complexity of the attack is $\mathcal{O}\big(23^3 47^3 2^{3\times 22)}\big) \approx 2^{96.2}$, that is a gain around 2^{32} compared to the best attack considered in [11] and about 2^{20} compared to the best attack found in [17, Subsec. 3.2].

The third case is the most interesting. Here we can freely choose the dimension of the projected code. Keep in mind that we want the Gilbert-Varshamov bound greater than d which is the case when the dimension k" of the projected code is $\geqslant 4$. We choose $k" = 4$ and we have in this case an attack of complexity $2^{43.6}$ which clearly leads to a feasible attack.

In [13], a new set of parameters is proposed, as follows :

n	k	m	q	d	security
82	41	41	2	5	80
106	53	53	2	6	128
74	37	23	2^4	4	100

In each case the factorization of $X^k - 1$ in $\mathbb{F}_q[X]$ is given by
(i) $X^{41} - 1 = (X-1)PQ$ with $\deg P = \deg Q = 20$
(ii) $X^{53} - 1 = (X-1)\sum_{i=0}^{52} X^i$
(iii) $X^{37} - 1 = (X-1)P_1 \dots P_4$ with $\deg P_i = 9$, for all $i \in [\![1; 4]\!]$.

The first case allow a non-trivial projection but it is not sufficient to obtain a better complexity than 80. In the second case, we can only use the folding attack, and its complexity is greater than 128.

In the third case, we can choose a projected code of dimension 9 and we have an attack of complexity $2^{87,1}$, that is a gain around 2^{13}.

As we can see, it is crucial to choose k such that $X^k - 1$ has the minimum of factors in $\mathbb{F}_q[X]$, it can always be factorisable in $(X-1)(\sum_{i=0}^{k-1} X^i)$ so one have

to choose $\sum_{i=0}^{k-1} X^i$ irreducible in $\mathbb{F}_q$. This implies k prime but it is not sufficient, as the third case of the parameters proposed in [13] proves it.

References

[1] Anja Becker, Antoine Joux, Alexander May, and Alexander Meurer. Decoding random binary linear codes in $2^{n/20}$: How $1 + 1 = 0$ improves information set decoding. In *Advances in Cryptology - EUROCRYPT 2012*, Lecture Notes in Comput. Sci. Springer, 2012.

[2] Elwyn Berlekamp, Robert McEliece, and Henk van Tilborg. On the inherent intractability of certain coding problems. *IEEE Trans. Inform. Theory*, 24(3):384–386, May 1978.

[3] Jonathan F. Buss, Gudmund S. Frandsen, and Jeffrey O. Shallit. The computational complexity of some problems of linear algebra. *J. Comput. System Sci.*, 58(3):572–596, June 1999.

[4] Florent Chabaud and Jacques Stern. The cryptographic security of the syndrome decoding problem for rank distance codes. In *Advances in Cryptology - ASIACRYPT 1996*, pages 368–381, Kyongju, Korea, November 1996.

[5] Philippe Delsarte. Bilinear forms over a finite field, with applications to coding theory. *J. Comb. Theory, Ser. A*, 25(3):226–241, 1978.

[6] Jean-Charles Faugère, Françoise Levy-dit Vehel, , and Ludovic Perret. Cryptanalysis of Minrank. In David Wagner, editor, *Advances in Cryptology - CRYPTO 2008*, volume 5157 of *Lecture Notes in Comput. Sci.*, pages 280–296, 2008.

[7] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. Structural weakness of compact variants of the McEliece cryptosystem. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT 2014*, pages 1717–1721, Honolulu, HI, USA, July 2014.

[8] Jean-Charles Faugère, Ayoub Otmani, Ludovic Perret, Frédéric de Portzamparc, and Jean-Pierre Tillich. Structural cryptanalysis of McEliece schemes with compact keys. *Des. Codes Cryptogr.*, 2015. to appear, see also IACR Cryptology ePrint Archive, Report2014/210.

[9] Ernst M. Gabidulin, A. V. Paramonov, and O. V. Tretjakov. Ideals over a non-commutative ring and their applications to cryptography. In *Proceedings of the Workshop on the Theory and Application of Cryptographic Techniques*, number 547 in LNCS, pages 482–489, Brighton, April 1991.

[10] Ernst M. Gabidulin and Nina I. Pilipchuk. Symmetric matrices and codes correcting rank errors beyond the $(d-1)/2$ bound. *Discrete applied Math.*, 154(2):305–312, 2006.

[11] Philippe Gaborit, Gaétan Murat, Olivier Ruatta, and Gilles Zémor. Low rank parity check codes and their application to cryptography. In *Proceedings of the Workshop on Coding and Cryptography WCC'2013*, Bergen, Norway, 2013. Available on `www.selmer.uib.no/WCC2013/pdfs/Gaborit.pdf`.

[12] Philippe Gaborit, Olivier Ruatta, and Julien Schrek. On the complexity of the rank syndrome decoding problem. *CoRR*, abs/1301.1026, 2013.

[13] Philippe Gaborit, Olivier Ruatta, Julien Schrek, and Gilles Zémor. New results for rank-based cryptography. In *Progress in Cryptology - AFRICACRYPT 2014*, volume 8469 of *Lecture Notes in Comput. Sci.*, pages 1–12, 2014.

[14] Aviad Kipnis and Adi Shamir. Cryptanalysis of the HFE public key cryptosystem by relinearization. In *Advances in Cryptology - CRYPTO'99*, volume 1666 of *Lecture Notes in Comput. Sci.*, pages 19–30, Santa Barbara, California, USA, August 1999. Springer.

[15] Kristine Lally and Patrick Fitzpatrick. Algebraic structure of quasicyclic codes. *Discrete applied Math.*, 111(1):157–175, 2001.

[16] San Ling and Patrick Solé. On the algebraic structure of quasi-cyclic codes. I. finite fields. *IEEE Trans. Inform. Theory*, 47(7):2751–2760, 2001.

[17] Pierre Loidreau. On cellular code and their cryptographic applications. In I. Landjev G. Kabatiansky, editor, *Proceedings of ACCT14 (algebraic and combinatorial coding theory)*, pages 234–239, Svetlogorsk, Russia, September 2014.

[18] Pierre Loidreau and Nicolas Sendrier. Weak keys in the McEliece public-key cryptosystem. *IEEE Trans. Inform. Theory*, 47(3):1207–1211, 2001.

[19] Robert J. McEliece. *A Public-Key System Based on Algebraic Coding Theory*, pages 114–116. Jet Propulsion Lab, 1978. DSN Progress Report 44.

[20] Rafael Misoczki, Jean-Pierre Tillich, Nicolas Sendrier, and Paulo S. L. M. Barreto. MDPC-McEliece: New McEliece variants from moderate density parity-check codes. In *Proc. IEEE Int. Symposium Inf. Theory - ISIT*, pages 2069–2073, 2013.

[21] Alexei V. Ourivski and Thomas Johansson. New technique for decoding codes in the rank metric and its cryptography applications. *Problems of Information Transmission*, 38(3):237–246, 2002.

[22] Raphael Overbeck. A new structural attack for GPT and variants. In *Mycrypt*, volume 3715 of *LNCS*, pages 50–63, 2005.